Support Systems Analysis

Drop Support Systems Analysis

Date: 2026-02-22 **Author:** John (AI Director) **Status:** MVP Hardening Phase (0.5) **Purpose:** Comprehensive analysis of support systems for production-ready fintech deployment

---

Executive Summary

Drop currently has **foundational support systems** in place but requires **critical enhancements** before production launch. The application has health checks, CI/CD, error tracking (client-side), and basic alerting, but lacks enterprise-grade observability, audit logging, and incident response procedures required for a PSD2-compliant fintech service.

Key Findings:

- ✅ **Strong foundation:** Comprehensive CI/CD with >80% coverage, health checks, structured logging
- ⚠️ **Critical gaps:** No server-side error tracking, no audit trails, no APM, limited incident response
- 🚨 **Production blockers:** 6 P0 items must be addressed before go-live (see Gap Analysis)

Recommendation: Implement P0 systems immediately (est. 2-3 days), defer P1 to Phase 2 (banking integration), and P2 to post-launch optimization.

Current State

1. Monitoring — Uptime & Health Checks

What Exists

- ✅ **Health endpoint:** `/api/health` with database connectivity verification
  - Checks: DB query latency, driver type (pg/sqlite), service mode, uptime
  - Returns: `ok` (200), `degraded` (200), or `down` (503)

  - Source: `src/drop-app/src/app/api/health/route.ts`
- ✅ **Container health checks:**
  - Docker: 30s interval, 10s timeout, 3 retries
  - Fly.io: 30s interval, 10s grace period, 5s timeout
  - Auto-restart on failure
- ✅ **External uptime monitoring (ready to deploy):**
  - BetterStack setup guide documented
  - Free tier: 10 monitors, 3-min interval, SMS/email/Slack alerts
  - Documentation: `docs/infrastructure/BETTERSTACK-SETUP.md`
- ✅ **Cron health check script:**
  - `infrastructure/health-check.sh` — AWS App Runner endpoint
  - Slack webhook integration (optional)
  - Can run via cron for local monitoring

What's Missing

- ❌ **Synthetic monitoring:** No transaction flow testing (login → send money → verify)
- ❌ **Multi-region checks:** No geographic availability testing
- ❌ **SLA tracking:** No uptime percentage calculation or reporting
- ❌ **Dependency monitoring:** No checks for external services (Swan API, BankID, Sumsub)

Assessment

Status: Adequate for MVP, requires enhancement for production. **Gap:** External monitoring configured but not deployed. Synthetic checks needed.

---

2. Logging — Centralized Log Aggregation

What Exists

- ✅ **Structured logging:**
  - JSON format with timestamp, level, message, requestId, metadata
  - Source: `src/drop-app/src/lib/logger.ts`
  - Writes to stdout (Docker-friendly)
- ✅ **Request correlation:**
  - `x-request-id` header extraction or UUID generation
  - Request context propagation through logger instances
- ✅ **Log levels:** debug, info, warn, error

What's Missing

- ❌ **Log aggregation:** Logs write to stdout but aren't collected or indexed
- ❌ **Log retention:** No policy for how long logs are kept
- ❌ **Log search:** No way to query logs across time/instances

- ❌ **Log forwarding:** No integration with log management service
- ❌ **Sensitive data scrubbing:** Logger doesn't automatically redact PII

Assessment

Status: Foundation exists, but logs are ephemeral (lost on container restart). **Gap:** Critical for incident investigation and compliance audits. Need CloudWatch Logs or similar.

---

3. Error Tracking — Error Capture & Alerting

What Exists

- ✅ **Client-side error tracking:**
  - Sentry browser integration (`@sentry/browser`)
  - PII scrubbing (passwords, pins, card numbers, fødselsnummer)
  - 10% trace sampling for performance monitoring
  - Source: `src/drop-app/src/lib/sentry.ts`, `SENTRY.md`
- ✅ **Error spike detection:**
  - Tracks errors in rolling 1-minute window
  - Alerts when >5 errors in 60 seconds
  - Source: `src/drop-app/src/lib/alerts.ts:trackError()`
- ✅ **Global error boundaries:**
  - React error boundaries for component crashes
  - `global-error.tsx` catches unhandled errors

What's Missing

- ❌ **Server-side error tracking:** Sentry removed from server due to Next.js 16 Turbopack incompatibility (MC #1271)
- ❌ **API error context:** Server errors log to console only, no structured capture
- ❌ **Error attribution:** Can't trace errors to specific users or transactions
- ❌ **Error deduplication:** Same error reported multiple times clogs alerts

Assessment

Status: Client errors tracked, server errors blind. **Gap:** CRITICAL — server-side errors (API, DB, integrations) are invisible. P0 fix required.

---

4. Alerting — On-Call & Escalation

What Exists

- ✅ **Slack alerting:**

- Operational alerts with severity levels (info/warning/critical)
- 10-minute cooldown per alert title (spam prevention)
- Source: `src/drop-app/src/lib/alerts.ts`

- **Lifecycle alerts:**
  - App startup notification
  - Graceful shutdown notification
  - Source: `instrumentation.ts`
- **Error spike alerts:**
  - Automatic critical alert when >5 errors/minute

What's Missing

- **On-call rotation:** No defined on-call schedule or escalation policy
- **Alert routing:** All alerts go to same Slack channel, no severity-based routing
- **Alert escalation:** No automatic escalation after N minutes of unresolved incident
- **Alert acknowledgment:** Can't mark alerts as "acknowledged" or "resolved"
- **SMS/phone alerts:** Critical incidents only notify via Slack (single point of failure)
- **Alert testing:** No way to test alert pipeline without triggering real incidents

Assessment

Status: Basic alerting works for small team, inadequate for 24/7 production. **Gap:** Need on-call schedule, escalation policy, and multi-channel delivery.

---

5. Security Monitoring — WAF, DDoS, Anomaly Detection, Audit Logs

What Exists

- **WAF rules defined:**
  - CSRF origin validation (implemented in middleware)
  - Rate limiting on auth endpoints (10 req/60s)
  - CSP headers with nonce-based script loading
  - Source: `infrastructure/waf-rules.md`, `src/drop-app/src/middleware.ts`
- **Container security scanning:**
  - Trivy vulnerability scanner in CI/CD
  - Blocks HIGH/CRITICAL vulnerabilities
  - SARIF upload to GitHub Security tab
- **Dependency scanning:**
  - `npm audit` in CI pipeline (prod deps only)
- **AML transaction monitoring:**
  - 5 automated rules: structuring, velocity, high amount, high-risk corridor, unusual pattern
  - Alerts stored in `aml_alerts` table

- Source: `src/drop-app/src/lib/transaction-monitor.ts`

What's Missing

- ❌ **WAF deployment:** Rules defined but not deployed (requires CDN/reverse proxy)
- ❌ **DDoS protection:** No rate limiting at network edge, only app-level
- ❌ **Intrusion detection:** No IDS/IPS monitoring unusual access patterns
- ❌ **Audit logs:** No immutable log of authentication, authorization, data access events (PSD2 requirement)
- ❌ **Security incident response plan:** No runbook for security breaches
- ❌ **Penetration testing:** No external security audit completed

Assessment

Status: Security-aware codebase, but monitoring/audit infrastructure missing. **Gap:** CRITICAL — audit logs are PSD2/GDPR compliance requirement. P0 fix.

---

6. Performance — APM, Latency Tracking, Resource Utilization

What Exists

- ✅ **Health check latency:**
  - DB query time measured in health endpoint
  - Reported in milliseconds
- ✅ **Performance budgets in CI:**
  - Coverage thresholds enforced (80/70/80/80)

What's Missing

- ❌ **APM (Application Performance Monitoring):** No distributed tracing
- ❌ **API latency tracking:** Don't know which endpoints are slow
- ❌ **Database performance:** No slow query alerts or query profiling
- ❌ **Resource utilization:** No CPU/memory/disk usage monitoring
- ❌ **Frontend performance:** No Core Web Vitals tracking (LCP, FID, CLS)
- ❌ **Transaction timing:** Can't measure end-to-end payment latency

Assessment

Status: Minimal. Can detect total outage but not performance degradation. **Gap:** Need before production to identify bottlenecks and capacity issues.

---

7. Database — Backups, Replication, Monitoring

What Exists

- ✅ **Automated backups (RDS):**
  - Daily automated snapshots, 7-day retention
  - Point-in-time recovery within 7 days
  - Source: `docs/dr-runbook.md`
- ✅ **Multi-AZ (production):**
  - RDS configured for high availability (if enabled)
- ✅ **Database health check:**
  - `SELECT 1` query in health endpoint verifies connectivity

What's Missing

- ❌ **Backup verification:** Snapshots created but never tested for restore
- ❌ **Backup monitoring:** No alerts if backup fails
- ❌ **Replication lag monitoring:** No alerts if replica falls behind
- ❌ **Connection pool monitoring:** No visibility into connection usage
- ❌ **Query performance:** No slow query log analysis
- ❌ **Storage monitoring:** No alerts before disk fills up

Assessment

Status: Basic backup/restore exists, monitoring gaps. **Gap:** Backup testing and proactive monitoring needed before production.

---

8. Incident Response — Runbooks, Status Page, Communication Plan

What Exists

- ✅ **DR runbook:**
  - Procedures for App Runner down, RDS down, full redeploy
  - Environment variable checklist
  - Contact escalation (John → Alem)
  - Source: `docs/dr-runbook.md`
- ✅ **Incident checklist:**
  - 8-step incident response workflow
  - Post-mortem requirement (48h)

What's Missing

- ❌ **Status page:** No public/customer-facing status page

- ❌ **Incident templates:** No standardized incident report format
- ❌ **Communication plan:** No templates for customer notifications during outages
- ❌ **Runbook coverage:** Only covers infrastructure, missing:
  - Payment failures (PISP/AISP errors)
  - BankID integration issues
  - KYC/AML false positive handling
  - Data breach response
- ❌ **Runbook testing:** Procedures documented but never executed

Assessment

Status: Basic DR runbook exists, lacks fintech-specific scenarios. **Gap:** Need payment/banking integration runbooks before Phase 2.

---

9. CI/CD — Build Pipeline, Deployment, Rollback

What Exists

- ✅ **Comprehensive CI pipeline:**
  - Multi-package change detection
  - Lint, typecheck, unit tests, E2E (Playwright), mutation testing (Stryker)
  - Coverage thresholds enforced (80/70/80/80) with ratchet (never decrease)
  - Docker build + Trivy security scan
  - Quality gate (required status check)
  - Source: `.github/workflows/ci.yml`
- ✅ **Deployment workflows:**
  - GitHub Actions for deploy (backend, mobile)
  - Terraform for infrastructure
  - Source: `.github/workflows/deploy.yml`, `terraform-ci.yml`

What's Missing

- ❌ **Automated rollback:** Deployment failure doesn't auto-revert
- ❌ **Canary deployments:** All-or-nothing deployment, no gradual rollout
- ❌ **Deployment monitoring:** No automatic health check after deploy
- ❌ **Deployment notifications:** Team not notified of deployments/failures
- ❌ **Infrastructure drift detection:** Terraform state not continuously validated

Assessment

Status: Strong quality gate, weak deployment safety. **Gap:** Add post-deployment health checks and rollback automation.

---

10. Compliance — Audit Trails, Data Retention, GDPR/PSD2 Logging

What Exists

- ✅ **AML monitoring:**
  - Transaction alerts stored in `aml_alerts` table
  - 5 risk categories tracked
- ✅ **Security audit completed:**
  - 4 CRITICAL, 5 HIGH, 6 MEDIUM, 4 LOW findings documented
  - Source: `security/drop-security-rapport.md`
- ✅ **Data retention service:**
  - Code exists for GDPR compliance
  - Source: `src/drop-app/src/lib/services/data-retention.ts`

What's Missing

- ❌ **Audit logs:** No immutable record of:
  - User authentication events (login, logout, failed attempts)
  - Authorization decisions (who accessed what, when)
  - Data modifications (user profile changes, transaction edits)
  - Administrative actions (KYC approvals, AML reviews)
- ❌ **Audit log retention policy:** PSD2 requires 5+ years
- ❌ **Audit log integrity:** No cryptographic proof of non-tampering
- ❌ **Compliance reporting:** No automated report generation for regulators
- ❌ **STR (Suspicious Transaction Report) workflow:** AML alerts created but no submission process

Assessment

Status: CRITICAL GAP. Audit logs are PSD2 legal requirement. **Gap:** P0 — must implement before production launch.

Gap Analysis

P0 — Production Blockers (Must Fix Before Go-Live)

#	Category	Gap	Impact	Effort

1	**Error Tracking**	No server-side error monitoring	Can't detect/debug API failures	4h
2	**Compliance**	No audit logs (auth, data access, admin actions)	PSD2 non-compliance, legal risk	8h
3	**Security**	WAF rules defined but not deployed	Vulnerable to SQLi, XSS, DDoS	2h (config)
4	**Logging**	No log aggregation/retention	Can't investigate incidents	2h (CloudWatch setup)
5	**Monitoring**	BetterStack configured but not deployed	No external incident detection	1h (account setup)
6	**Incident Response**	No payment/banking failure runbooks	Can't recover from PISP/BankID outages	4h

Total P0 effort: ~21 hours (2-3 days)

P1 — Needed Soon (Before Phase 2: Banking Integration)

#	Category	Gap	Impact	Effort
7	**Alerting**	No on-call rotation or escalation policy	Incidents may go unnoticed outside work hours	2h
8	**Performance**	No APM for distributed tracing	Can't diagnose slow transactions	4h
9	**Database**	No backup testing or monitoring	Backups may be corrupt, undetected	3h
10	**Security**	No penetration testing	Unknown vulnerabilities	16h (external)
11	**CI/CD**	No automated rollback on deployment failure	Bad deploys cause extended outages	6h
12	**Compliance**	No STR submission workflow	Can't fulfill AML obligations	8h

Total P1 effort: ~39 hours (5 days)

P2 — Nice to Have (Post-Launch Optimization)

#	Category	Gap	Impact	Effort
13	**Monitoring**	No synthetic transaction monitoring	Can't detect broken user flows	8h
14	**Performance**	No Core Web Vitals tracking	Poor user experience undetected	4h
15	**Alerting**	No SMS/phone alerts for critical incidents	Slack outage = missed alerts	2h
16	**Database**	No slow query alerts	Performance degradation undetected	6h
17	**Security**	No IDS/IPS for intrusion detection	Advanced attacks undetected	16h
18	**Incident Response**	No public status page	Customers unaware of outages	4h

Total P2 effort: ~40 hours (5 days)

Implementation Plan

Phase 1: P0 Production Blockers (NOW — before Phase 1 demo)

Goal: Address legal/compliance requirements and critical observability gaps.

1.1 Server-Side Error Tracking (4h)

Problem: All server errors invisible after Sentry removed (Next.js 16 Turbopack incompatibility).

Solution:

- **Option A:** Sentry Edge SDK (compatible with Next.js middleware)
  - Install: `@sentry/nextjs` with edge-only config
  - Capture server errors via `captureException()` in middleware
  - Source maps via Sentry webpack plugin
- **Option B:** Custom error aggregation service
  - POST errors to internal `/api/errors/capture` endpoint
  - Store in `error_logs` table with context
  - Alert on spike detection

Deliverable:

- `src/drop-app/sentry.edge.config.ts` (if Option A)
- Updated `src/drop-app/src/lib/sentry-server.ts` with edge-compatible capture
- Test: Trigger 500 error, verify Sentry event created

Files: `infrastructure/error-tracking-setup.md`

1.2 Audit Logging System (8h)

Problem: PSD2 requires immutable audit trail for auth, data access, admin actions.

Solution:

- Create `audit_logs` table:

```
CREATE TABLE audit_logs (
  id TEXT PRIMARY KEY,
  timestamp TEXT NOT NULL,
  user_id TEXT,
  action TEXT NOT NULL, -- 'login', 'data_access', 'kyc_approval', etc.
  resource_type TEXT, -- 'user', 'transaction', 'aml_alert'
  resource_id TEXT,
  metadata JSON,
  ip_address TEXT,
  user_agent TEXT,
  request_id TEXT,
  result TEXT -- 'success', 'failure', 'denied'
);
CREATE INDEX idx_audit_user ON audit_logs(user_id, timestamp);
CREATE INDEX idx_audit_action ON audit_logs(action, timestamp);
```

- Audit functions:

```
auditLog({
  userId: 'usr_123',
  action: 'login_success',
  resourceType: 'user',
  resourceId: 'usr_123',
  metadata: { method: 'bankid' },
  ip: '1.2.3.4',
  userAgent: 'Mozilla...',
  requestId: 'req_456'
});
```

- Integrate at:
  - `POST /api/auth/login` (login_success, login_failure)
  - `POST /api/auth/logout` (logout)
  - `GET /api/users/:id` (data_access)
  - `PATCH /api/users/:id/kyc` (kyc_approval, kyc_rejection)
  - `PATCH /api/aml-alerts/:id` (aml_review)

Deliverable:

- `src/drop-app/src/lib/audit-log.ts` (audit logging functions)
- Migration: `migrations/003_audit_logs.sql`
- Integration in auth routes and admin endpoints
- Retention policy: Document 5-year retention for PSD2 compliance

Files: `support/audit-logging-setup.md`

1.3 WAF Deployment (2h)

Problem: WAF rules defined but not enforced (requires reverse proxy).

Solution:

- **Option A:** Cloudflare WAF (recommended)
  - Already using Cloudflare for DNS (terraform module exists)
  - Free tier includes basic WAF rules
  - Configure: SQLi, XSS, path traversal rules from `infrastructure/waf-rules.md`
- **Option B:** AWS WAF (if using App Runner directly)
  - $5/month + $1/million requests
  - Associate with App Runner service

Deliverable:

- Cloudflare WAF configuration (Terraform or UI)
- Test: Send SQLi payload, verify 403 response
- Document: Update `infrastructure/waf-rules.md` with deployment steps

Files: `infrastructure/cloudflare-waf-setup.md`

1.4 Log Aggregation (2h)

Problem: Structured logs write to stdout but aren't retained or searchable.

Solution:

- **AWS CloudWatch Logs** (App Runner auto-integrates):
  - App Runner streams stdout → CloudWatch Logs automatically

    - Configure retention: 30 days (production), 7 days (staging)
    - Set up log insights queries for common patterns
- **Fly.io (staging):**
    - `fly logs` stores last 24h by default
    - Optional: Forward to external service (Papertrail, Logtail)

Deliverable:

- CloudWatch Logs retention policy configured
- Log Insights queries:
    - All errors: `fields @timestamp, message | filter level = "error"`
    - User actions: `fields @timestamp, userId, message | filter userId = "usr_123"`
    - Request trace: `fields @timestamp, requestId, message | filter requestId = "req_456"`
- Documentation: `infrastructure/logging-setup.md`

Files: `infrastructure/cloudwatch-logs-setup.md`

1.5 External Uptime Monitoring (1h)

Problem: BetterStack documented but not deployed.

Solution:

- Sign up: https://betterstack.com/uptime (free tier)
- Create monitors:
    1. **Production health:** `https://9ef3szvvsb.eu-west-1.awsapprunner.com/api/health`
        - Interval: 3 minutes
        - Keyword check: `"status":"ok"`
    2. **Staging health:** `https://drop-staging.fly.dev/api/health`
    3. **Landing page:** `https://getdrop.no` (when live)
- Slack integration: Connect to `#drop-ops` channel
- Email alerts: `alem@alai.no`

Deliverable:

- BetterStack account with 3 monitors configured
- Test: Pause monitor, verify alert received
- Documentation: Update `docs/infrastructure/BETTERSTACK-SETUP.md` with credentials

Files: `support/betterstack-deployment.md`

1.6 Payment/Banking Failure Runbooks (4h)

Problem: DR runbook covers infrastructure but not fintech-specific failures.

Solution:

- Create runbooks for:
  1. **BankID integration failure** (authentication blocked)
  2. **PISP payment failure** (remittance/QR payment rejected)
  3. **AISP balance retrieval failure** (can't fetch account balance)
  4. **Swan API outage** (BaaS provider down)
  5. **Sumsub KYC failure** (identity verification unavailable)
  6. **Neonomics open banking outage**
- Each runbook includes:
  - Symptoms (what users see)
  - Diagnosis steps (check service status, logs, error codes)
  - Recovery procedure (fallback, retry, escalation)
  - Customer communication template

Deliverable:

- `support/runbooks/bankid-failure.md`
- `support/runbooks/pisp-payment-failure.md`
- `support/runbooks/aisp-balance-failure.md`
- `support/runbooks/swan-api-outage.md`
- `support/runbooks/sumsub-kyc-failure.md`
- `support/runbooks/neonomics-outage.md`

Files: Created in `/Users/makinja/ALAI/products/Drop/support/runbooks/`

Phase 2: P1 Items (Phase 2: Banking Integration)

Defer to Phase 2 when real banking integrations are live and need production-grade support.

Priority order:

1. Penetration testing (external security audit)
2. APM for transaction tracing (identify slow payments)
3. On-call rotation and escalation policy
4. Automated rollback on failed deployments
5. Backup testing and monitoring
6. STR submission workflow (AML compliance)

Phase 3: P2 Items (Post-Launch)

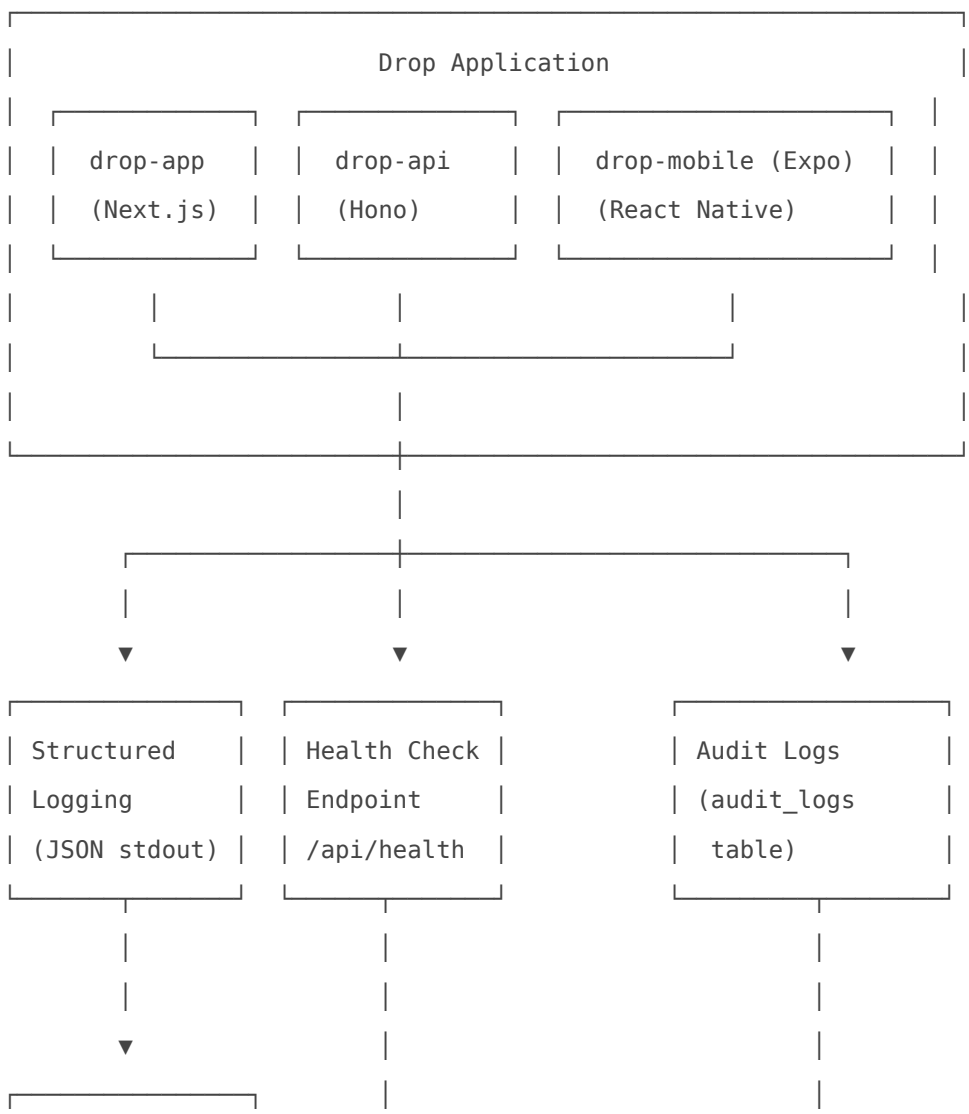
Optimize after initial production deployment and user feedback.

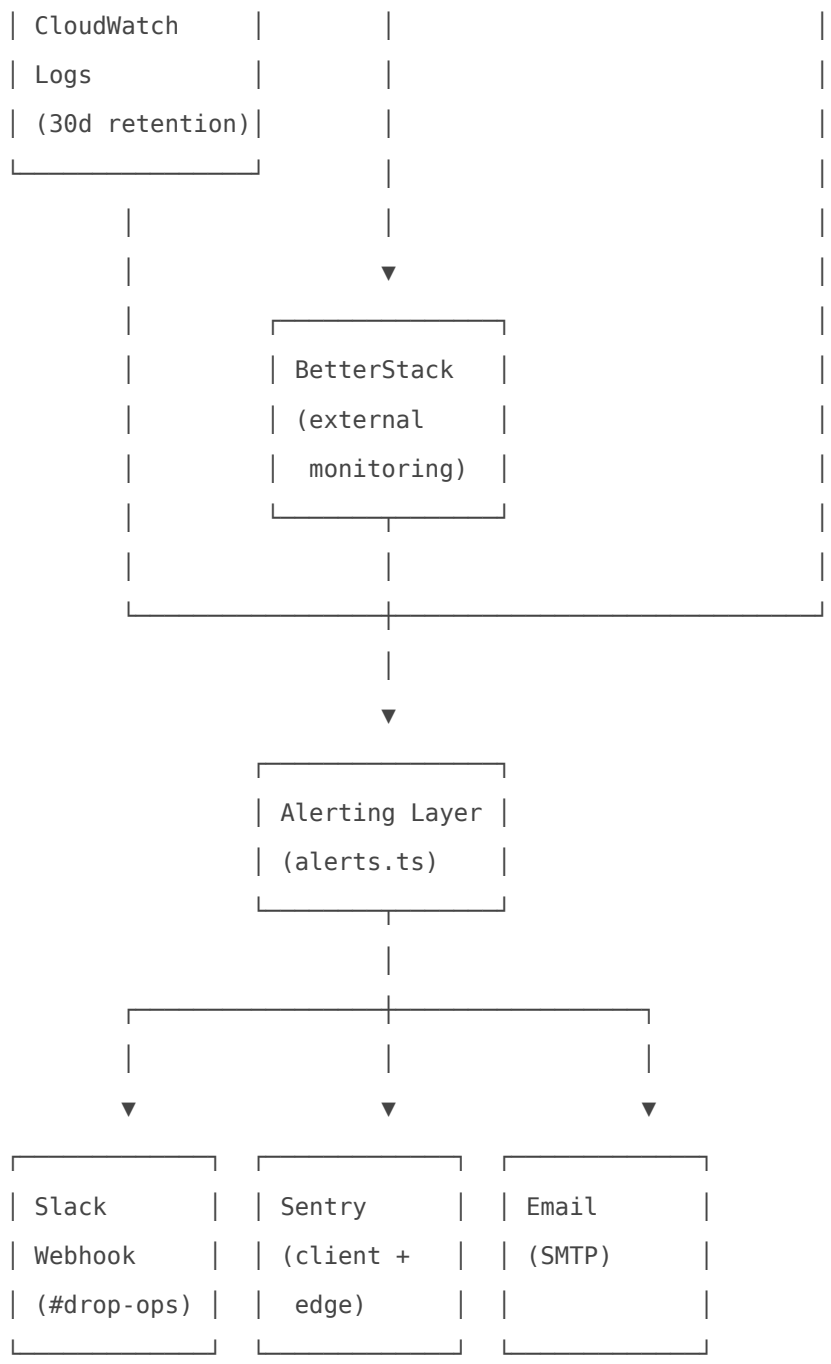
Priority order:

1. Synthetic transaction monitoring (test critical user flows)
2. Public status page (customer transparency)
3. Core Web Vitals tracking (frontend performance)
4. SMS/phone alerts (redundancy)
5. Slow query monitoring (database optimization)
6. IDS/IPS (advanced threat detection)

Architecture

Support Systems Connectivity





Data Flows

1. **Error Flow:**
   - Client error → Sentry browser → Slack alert (if spike)
   - Server error → Sentry edge → CloudWatch Logs → Slack alert
   - API 5xx → `trackError()` → Spike detection → Slack
2. **Monitoring Flow:**
   - App → stdout → CloudWatch Logs
   - App → `/api/health` → BetterStack → Slack/Email/SMS
   - Container → Docker health check → Auto-restart
3. **Audit Flow:**
   - User action → `auditLog()` → `audit_logs` table

- Compliance query → SQL export → Regulator submission

4. **Incident Flow:**
   - Alert → Slack `#drop-ops`
   - Unacknowledged (5 min) → Email to Alem
   - Unresolved (15 min) → SMS (BetterStack escalation)
   - Incident → Runbook → Recovery → Post-mortem

Cost Estimate

Free Tier (MVP)

- □ CloudWatch Logs: 5 GB ingestion/month free (AWS Free Tier)
- □ BetterStack: 10 monitors, 3-min interval, unlimited alerts
- □ Sentry: 5K events/month free
- □ GitHub Actions: 2000 minutes/month free
- □ Terraform state: S3 free tier (first 12 months)

Total MVP cost: $0/month

Paid Services (Production)

- CloudWatch Logs: ~$5/month (30 GB ingestion estimate)
- BetterStack Pro: $20/month (30s interval, SMS alerts)
- Sentry Team: $26/month (50K events, enhanced features)
- **Optional:** Datadog APM: $15/host/month (~$45 for 3 hosts)

Total production cost: ~$50-100/month (without APM)

Recommendations

Immediate (This Week)

1. □ **Deploy BetterStack** (1h) — External monitoring is fast win
2. □ **Configure CloudWatch retention** (30 min) — Logs already flow, just set policy
3. □ **Create audit log schema** (2h) — Start with table, integrate incrementally

Before Phase 1 Demo (Next 2 Weeks)

4. 🔴 **Implement server-side error tracking** (4h) — Sentry edge or custom
5. 🔴 **Write payment failure runbooks** (4h) — Prepare for demo questions
6. 🔴 **Deploy Cloudflare WAF** (2h) — Security hygiene

Before Phase 2 Go-Live (Next 2-3 Months)

7. 🟡 **External penetration test** (hire security firm, ~$5K budget)
8. 🟡 **APM implementation** (Datadog or Sentry Performance)
9. 🟡 **On-call rotation** (define schedule, test escalation)
10. 🟡 **Backup testing** (restore from snapshot, verify data integrity)

Post-Launch Optimization

11. 🟢 **Synthetic monitoring** (Checkly or custom Playwright tests)
12. 🟢 **Public status page** (BetterStack included, just enable)
13. 🟢 **Core Web Vitals** (Google Lighthouse CI integration)

Success Metrics

Before Go-Live (P0 Checklist)

- ☐ Server errors visible in Sentry (test: trigger 500, verify event)
- ☐ Audit logs capture login/logout (test: log in, check `audit_logs` table)
- ☐ WAF blocks SQLi attack (test: `?id=1' OR '1'='1`, expect 403)
- ☐ CloudWatch Logs retain 30 days (verify retention policy)
- ☐ BetterStack alerts on downtime (test: stop app, receive alert <5 min)
- ☐ Runbooks tested (simulate BankID failure, follow procedure)

Production KPIs

- **Uptime:** >99.9% (measured by BetterStack)
- **MTTD (Mean Time To Detect):** <3 minutes (external monitoring interval)
- **MTTR (Mean Time To Recover):** <15 minutes (via runbooks)
- **Error rate:** <0.1% of requests (tracked via Sentry)
- **Log retention:** 100% compliance (30 days CloudWatch, 5 years audit logs)
- **Alert noise:** <5 false positives/week (cooldown + severity tuning)

Appendices

A. Related Documentation

- `docs/infrastructure/MONITORING.md` — Current monitoring setup
- `docs/infrastructure/BETTERSTACK-SETUP.md` — External monitoring guide
- `docs/dr-runbook.md` — Infrastructure disaster recovery
- `infrastructure/waf-rules.md` — WAF rule definitions
- `security/drop-security-rapport.md` — Security audit findings

B. External Services

- BetterStack: https://betterstack.com/uptime
- Sentry: https://sentry.io/
- AWS CloudWatch: https://console.aws.amazon.com/cloudwatch/
- Cloudflare: https://dash.cloudflare.com/

C. Change History

- 2026-02-22: Initial analysis (John)

Next Actions:

1. Review this analysis with Alem
2. Approve P0 implementation plan
3. Begin P0 work (estimated 21 hours / 2-3 days)
4. Track progress in Mission Control tasks

Revision #8
Created 2026-02-23 11:29:19 UTC by John
Updated 2026-05-25 07:27:28 UTC by John